Compositional Verification of Interlocking Systems for Large Stations

Fantechi, Alessandro; Haxthausen, Anne Elisabeth; Macedo, Hugo Daniel dos Santos

Published in:
Software Engineering and Formal Methods

Link to article, DOI:
10.1007/978-3-319-66197-1_15

Publication date:
2017

Document Version
Peer reviewed version

Compositional Verification of Interlocking Systems for Large Stations

Alessandro Fantechi[1,2,⋆], Anne E. Haxthausen[1], and Hugo D. Macedo[1,3,⋆]

[1] DTU Compute, Technical University of Denmark, Lyngby, Denmark
aeha@dtu.dk
[2] DINFO, University of Florence, Firenze, Italy
alessandro.fantechi@unifi.it
[3] Department of Engineering, Aarhus University, Aarhus, Denmark
hdm@eng.au.dk

Abstract. Railway interlocking systems are responsible to grant exclusive access to a route, that is a sequence of track elements, through a station or a network. Formal verification that basic safety rules regarding exclusive access to routes are satisfied by an implementation is still a challenge for networks of large size due to the exponential computation time and resources needed.
Some recent attempts to address this challenge adopt a compositional approach, targeted to track layouts that are easily decomposable into sub-networks such that a route is almost fully contained in a sub-network: in this way granting the access to a route is essentially a decision local to the sub-network, and the interfaces with the rest of the network easily abstract away less interesting details related to the external world.
Following up on previous work, where we defined a compositional verification method that started considering routes that overlap between sub-networks in interlocking systems governing a multi-station line, we attack the verification of large networks, which are typically those in main stations of major cities, and where routes are very intertwined and can hardly be separated into sub-networks that are independent at some degree. At this regard, we study how the division of a complex network into sub-networks, using stub elements to abstract all the routes that are common between sub-networks, may still guarantee compositionality of verification of safety properties.

Keywords: railway interlocking, compositional verification, model checking

1 Introduction

Railway interlocking systems are those systems that are responsible to grant to a train the exclusive access to a *route*: a route is a sequence of track elements that are exclusively assigned for the movement of a train through a station or a

⋆ The authors' research conducted at DTU Compute was funded by Villum Fonden and by the RobustRailS project granted by Innovation Fund Denmark, respectively.

network. Granting of a route to a train occurs after a reservation request only if the track elements that form the route are not occupied by other trains, and if no conflicting route (that is, no other route that shares track elements with it) has been reserved by another train.

Errors in granting to a train the access to a route can obviously have catastrophic consequences; interlocking systems are therefore ranked as safety-critical systems, and this demands for high standards in the development of the software controlling interlocking systems. The standard CENELEC 50128 [2] labels such software with the highest *safety integrity level* (SIL4), and highly recommends the usage of formal methods and formal verification in its development process.

However, full formal verification of interlocking systems demands heavy if not infeasible computational resources for the phenomenon known as the state explosion problem, that is, the exponential growth of the state space with the number of elements in the controlled track layout. The most recent research in model checking and in applying model checking to the domain of railways [3,4,5,6,20,10,21] has developed techniques allowing the verification of models of the interlocking systems controlling quite large and complex networks. For example, abstraction techniques can be applied at the domain modelling level before the model checking is performed [10]. Other very efficient techniques applied to real world railway interlocking systems are bounded model checking [7] and k-induction [20].

However, formal verification that basic safety rules regarding exclusive access to routes are satisfied by an implementation is still a challenge for networks of very large size, due to the exponential computation time and resources needed.

Some recent attempts to address this challenge adopt a *compositional* approach, targeted to track layouts that are easily decomposable into sub-networks such that a route is almost fully contained in a sub-network: in this way granting the access to a route is essentially a decision local to the sub-network, and the interfaces with the rest of the network easily abstract away less interesting details related to the external world. This is the case of [11], where a station layout is divided into two symmetric components that can be separately verified using an assume-guarantee reasoning, and of our previous work [12,13], where we were able to divide a multi-station line into almost independent components by performing cuts in between the stations.

In this paper we extend our previous work [12,13] to provide a new, more complex, way of dividing large networks, such as those typically found in main stations of major cities, where routes are very intertwined and can hardly be separated into sub-networks that are independent. First, in section 2, we give a short introduction to the RobustRails verification tools we are using. Next, in section 3, we elaborate more on the idea of compositional verification and relate our new work to some past, related work. Then, in section 4 and section 5 we describe our compositional method with the new cut and make some experiments using it both for a smaller station and a large, real-world station. Finally, in section 6, some conclusions are drawn.

2 Interlocking Systems and Their Verification

In this section we briefly introduce the main notions of interlocking systems: we actually use the terminology and the assumptions of the new Danish ETCS Level 2 resignalling program, and we refer to [19,13] for a more detailed introduction. In this context, the specification of a given route-based interlocking system consists of two main components: (1) a railway network, and (2) a corresponding interlocking table.

A railway network consists of a number of track and track-side elements of different types[4]: linear sections, points, and marker boards. A *linear section* is a section with up to two neighbours: one in the *up* end, and one in the *down* end. For simplicity, in the examples and figures in the rest of this article, the *up* (*down*) direction is assumed to be the left-to-right (right-to-left) direction. A *point* can have up to three neighbours: one at the *stem*, one at the *plus* end, and one at the *minus* end. The *stem* and *plus* ends form the straight (main) path, and the *stem* and *minus* ends form the branching (siding) path. A point can be switched between two positions: PLUS and MINUS, selecting the main or siding paths, respectively. Linear sections and points are collectively called (train detection) sections, as they are provided with train detection equipment used by the interlocking system to detect the presence of trains. Along each linear section, up to two *marker boards* (one for each direction) can be installed. A marker board can only be seen in one direction and is used as reference location (for the start and end of routes) for trains going in that direction. There are no physical signals in ETCS Level 2, but interlocking systems have a *virtual signal* associated with each marker board. Train drivers do not visually see the aspect of virtual signals (OPEN or CLOSED), that is instead communicated to the onboard computer via a radio network. For simplicity, the terms *virtual signal*, *signal*, and *marker board* are used interchangeably throughout this paper.

A *route* is a path from a *source* signal to a *destination* signal in the given railway network. A route is called an *elementary route* if there are no signals that are located between its source signal and its destination signal, and that are intended for the same direction as the route. In railway signalling terminology, *setting* a route denotes the process of allocating the resources – i.e., sections, points, and signals – for the route, and then *locking* it exclusively for only one train when the resources are allocated.

An *interlocking table* specifies the elementary routes in the given railway network and the conditions for setting these routes. A route is defined by the following attributes:

- src(r) – the source signal of r,
- dst(r) – the destination signal of r,
- path(r) – the list of sections constituting r's path from src(r) to dst(r),
- points(r) – the required position of the points along the route r
- signals(r) – the required settings of signals
- conflicts(r) – a set of conflicting routes which must not be set while r is set.

[4] Here we only show types that are relevant for the work presented in this article.

Examples of network layouts and related interlocking tables are deferred to section 4.

Typical safety properties required of an interlocking system are that it always ensures the following safety conditions:

1. **No collisions:** Two trains must never occupy the same track section at the same time.
2. **No derailments:** A point must not be switched, while being occupied by a train.

All required safety properties are expressed as *generic* conditions leading to *specific* conditions for each specific case of a network. Notice that considering such typical safety properties, a route defines the maximal subset of elements whose status affects the safety property, that is, no element outside a route, or, at most, two conflicting routes, can affect a safety property for that route(s).[5]

The RobustRailS verification method [17,18,19,20] is a combination of formal methods and a domain-specific language (DSL) to express network diagrams and interlocking tables. A tool is provided by the RobustRailS environment to transform the DSL description into inputs to the model checker, that is, *i)* a behavioural model of the interlocking system and its environment, and *ii)* the required safety properties given as linear temporal logic formulae.

The RobustRailS tools can be used to verify the design of an interlocking system in the following steps:

1. A DSL specification of the configuration data (a network layout and its corresponding interlocking table) is constructed in the following order:
   (a) first the network layout,
   (b) and then the interlocking table (this is either done manually or generated automatically from the network layout).
2. The static checker verifies whether the configuration data is statically well-formed according to the static semantics [19] of the DSL.
3. The generators instantiate a generic behavioural model and generic safety properties with the well-formed configuration data to generate the model input of the model checker and the safety properties.
4. The generated model instance is then checked against the generated properties by the bounded model checker performing a k-induction proof.

The static checking in step (2) is intended to catch errors in the network layout and interlocking table, while the model checking in step (4) is intended to catch safety violations in the control algorithm of the instantiated model.

The tool chain associated with the method has been implemented using the RT-tester framework [14,16].

[5] The subset is sometimes extended with overlaps (buffer zones at the end of paths), and points or signals needed for flank protection, since this is sometimes required to protect tracks occupied by a train from another train not succeeding to brake in due space. We do not consider these extra protections in this paper, and we refer to [15,19] for details and for their modelling.

3 Compositionality

Interlocking systems typically exhibit a high degree of *locality*: if we consider a typical safety property desired for an interlocking system, e.g. that the same track element shall not be reserved by more than one train at a time, it is likely that this property is not influenced by a train moving on a distant, or parallel, track element. Locality of a safety property can be exploited for verification purposes, so limiting the state space on which to verify it. This principle has been exploited in [22] to define domain-oriented optimisation of the variable ordering in a BDD-based verification. Locality can be used also for slicing, as suggested in [3,9,8,1]: the idea is to consider only the portion of the model that has influence on the property to be verified, by a topological selection of interested track elements (therefore closely related to the *cone of influence* of the property): this allows for a much more efficient verification of the single property, but comes at the price of repeating the slicing and the verification for every property, and of separately checking that verifying slices does actually imply the satisfaction of desired properties for the whole system. Nevertheless, it appears that when automated, this process can offer significant time and memory savings.

Compositional verification of interlocking systems also exploits locality: the network layout is divided into two or more sub-networks, so that the separate verification of safety properties on the sub-networks can be used to prove safety properties on the whole network, with a significant advantage in terms of time and memory; moreover, the verification can be run once on the conjunction of all safety properties. Adopting a compositional approach actually needs a proof that the separate verification guarantees the safety properties for the whole network, and this proof depends on the type of division (from now on, *cut*) that is envisaged.

Indeed, in Figure 1 the dotted light green lines show three different cases of cuts of a network into two small networks:

a) A line connecting several stations is divided into sub-networks, each including a station; this case has been studied in our previous work [12,13]. We can see that the cut concerns a single track element, that behaves as source and destination of routes from/to one of the stations (A1u-L1u, A2u-L1u, L1d-A1d, L1d-A2d from/to station A, B1d-L1d, B2d-L1d, B3d-L1d, L1u-B1u, L1u-B2u, L1u-B3u from/to station B). Notice that all these routes are almost fully contained in one of the sub-networks: in this way granting the access to a route is essentially a decision local to the sub-network, apart from the single interface track element that is included in both routes of the sub-networks. This allows to consider, in the separate verification, the shared element as an abstraction of all the routes of the Station A when verifying properties related to Station B, and vice versa.
b) This is the case of a (possibly almost) symmetrical station, that is divided into two halves, as studied in [11]: the verification of one half takes into account assume/guarantee conditions at the interface with the other half. The

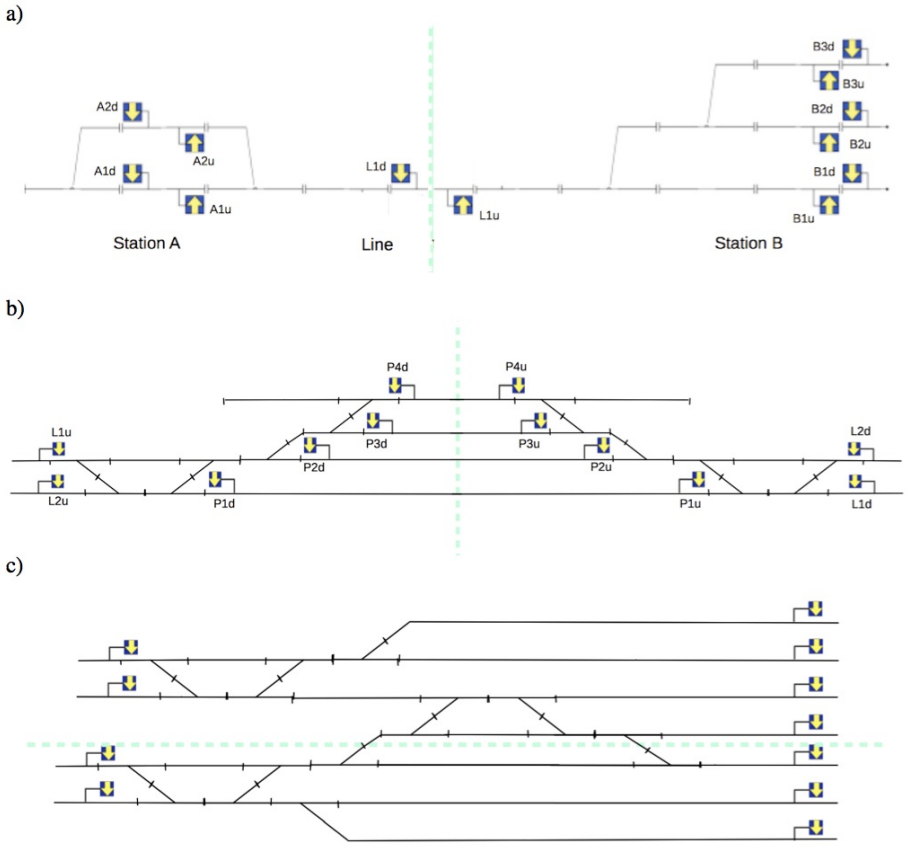

Fig. 1: Three example cuts.

verification effort is hence repeated for the two halves, with the extra effort of proving that assume/guarantee conditions do hold at the interface: locality allows such conditions to be rather simple so that they do not add much time to the verification. Again, a route is almost fully contained in one of the two sub-network, and the shared track elements act as an abstraction of the routes of the other sub-network, although there are several shared elements (e.g. routes L1u-P3u and L2u-P3u have a single track element shared with routes L1d-P3d and L2d-P3d).[6]

c) A more complex case is that of a (terminus) station where more lines (in this case, two double-track lines) converge from one side (in this case from the left side) to the station. The layout has to include paths allowing to go from any input track to any (or almost any) platform, through a sequence of points that in the figure goes from high tracks to low tracks, and vice versa. Notice that it is not possible to operate a cut like the ones of the

[6] For simplicity we consider here and in the following only the shortest path routes.

previous cases, since conflicting routes (in the layout in the figure only the markerboards in the up direction are reported) share more than one track element.
The dotted light green line shows a kind of cut, that we will call from now on *horizontal cut* to distinguish it from the previous cases, that looses the property that routes are almost fully contained in a sub-network. In the figure, e.g., routes L1u-P4u, L2u-P4u, L1u-P3u, L2u-P3u, from the high sub-network to the low sub-network share elements with all the routes having as destination P3u, P4u or P5u. This means that the kind of abstraction used in the previous cases can no longer be applied: in a sense, the layouts of cases a) and b) exhibit a *natural* place where to apply the cut, so that there are almost no interactions among the two parts, while the cut points in c) are chosen quite arbitrarily, just in order to reasonably decompose the network, but impacting on the middle elements of several routes.

The horizontal cut is the subject of the present paper, where we can show that a simple form of cut still allows for compositional verification. The research we are conducting aims in the end to come up with a subdivision process that exploits the characteristics of the network to provide a set of sub-networks, obtained with the most appropriate kind of cut, to be verified separately, so that safety of the whole layout can be deduced by the separate safety verification of the sub-network.

4 Horizontal Cut

In this section we explain how our compositional approach is done in three steps:

1. decomposition of the network into sub-networks using the horizontal cut,
2. decomposition of the interlocking table for the network by generation of interlocking tables for the sub-networks, and
3. safety verification for the sub-networks (from step 1) and their associated interlocking tables (from step 2).

Below, we also discuss and analyse the contents of the decomposed interlocking tables achieved in step 2 and examine the soundness of the approach. The explanations are done for an example network called ThreeTracksStation, but can easily be generalised to any network.

4.1 Decomposition of the Network

Fig. 2 shows first the layout of the ThreeTracksStation network. In this example we cut this network into two sub-networks by making a horizontal cut (the light green line) at the link between T9 and T10. To each of the two sub-networks (above and below the cut, respectively), two consecutive linear sections (which taken together form a *stub*) are added on the other side of the cut, in order to abstract the whole other sub-network. The new sections have proper markerboards

in order to satisfy network conditions about the positioning of markerboards at borders. The two resulting networks, called High and Low[7], are shown in Fig. 2.

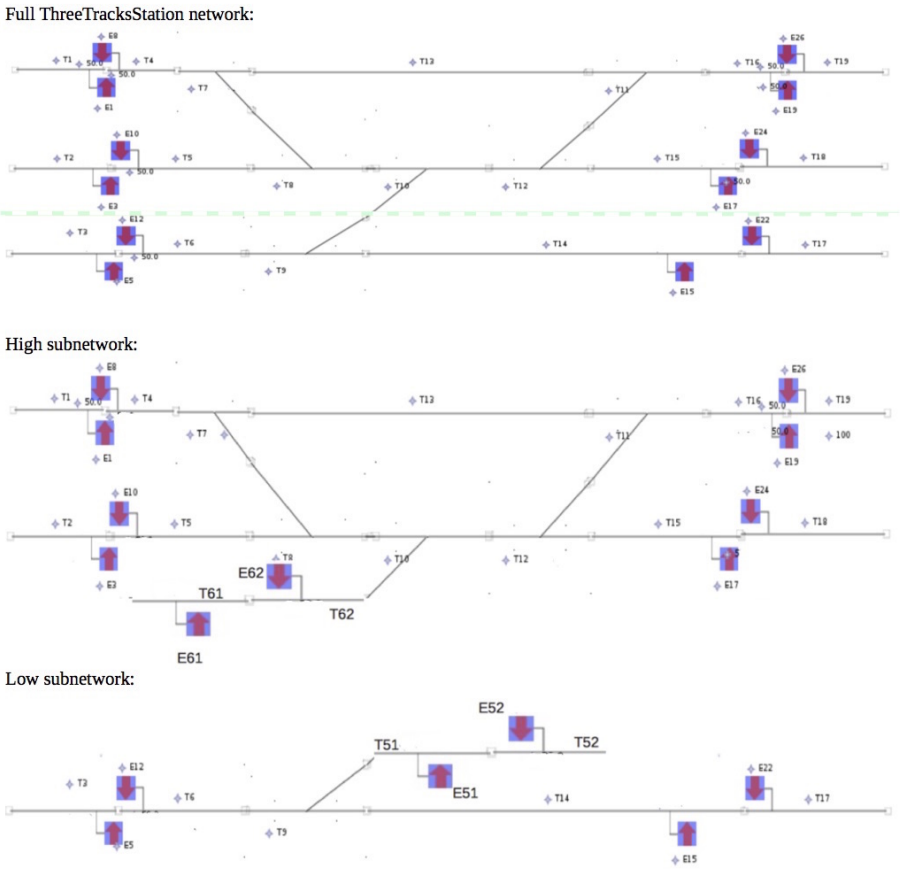

Fig. 2: Layouts of the ThreeTracksStation network and its sub-networks.

4.2 Decomposition of the Interlocking Table

In the second step, after having decomposed the network into the two sub-networks High and Low, the interlocking tables associated with these sub-networks are generated with the RobustRailS tool. The interlocking tables generated for the full network and for the two sub-networks are shown in Table 1.

[7] We reserve the words *up* and *down* for the train travel directions.

Table 1: The interlocking tables for the ThreeTracksStation, High and Low network of Fig.2.

				ThreeTracksStation		
id	**src**	**dst**	**path**	**points**	**signals**	**conflicts**
r1	E1	E19	T4;T7;T13;T11;T16	T7:p;T11:p	E26;E8	r12;r13;r1bis;r10;r5;r11;r5bis;r9;r14
r11	E1	E17	T4;T7;T8;T10;T12;T15	T10:p;T7:m;T8:m;T12:p	E24;E8	r1bis;r5;r12;r4;r5bis;r10;r9;r3;r6;r8;r14;r13;r1
r1bis	E1	E19	T4;T7;T8;T10;T12;T11;T16	T10:p;T7:m;T8:m;T11:m;T12:m	E26;E8	r8;r6;r13;r9;r3;r10;r5;r5bis;r14;r4;r12;r11;r1
r2	E22	E12	T14;T9;T6	T9:p	E15;E5	r4;r7;r12;r9;r8
r3	E24	E10	T15;T12;T10;T8;T5	T10:p;T8:p;T12:p	E17;E3	r6;r8;r4;r9;r10;r13;r14;r5bis;r12;r1bis;r11
r14	E24	E8	T15;T12;T10;T8;T7;T4	T10:p;T7:m;T8:m;T12:p	E1;E17	r4;r8;r13;r5bis;r10;r6;r9;r5;r12;r3;r1bis;r1;r11
r4	E24	E12	T15;T12;T10;T9;T6	T10:m;T9:m;T12:p	E17;E5	r6;r7;r10;r5bis;r13;r8;r12;r9;r2;r14;r3;r11;r1bis
r5	E26	E8	T16;T11;T13;T7;T4	T7:p;T11:p	E1;E19	r10;r9;r12;r13;r5bis;r11;r1;r1bis;r14
r13	E26	E10	T16;T11;T12;T10;T8;T5	T10:p;T8:p;T11:m;T12:m	E19;E3	r10;r5bis;r8;r9;r12;r6;r14;r1;r1bis;r3;r4;r5;r11
r5bis	E26	E8	T16;T11;T12;T10;T8;T7;T4	T10:p;T7:m;T8:m;T11:m;T12:m	E1;E19	r6;r8;r10;r12;r9;r14;r11;r13;r4;r3;r1bis;r1;r5
r12	E26	E12	T16;T11;T12;T10;T9;T6	T10:m;T9:m;T11:m;T12:m	E19;E5	r8;r10;r7;r6;r9;r1;r11;r5;r2;r5bis;r13;r4;r14;r3;r1bis
r6	E3	E17	T5;T8;T10;T12;T15	T10:p;T8:p;T12:p	E10;E24	r8;r9;r10;r3;r5bis;r4;r1bis;r14;r12;r11;r13
r10	E3	E19	T5;T8;T10;T12;T11;T16	T10:p;T8:p;T11:m;T12:m	E10;E26	r9;r8;r5;r13;r14;r6;r3;r5bis;r12;r1;r4;r11;r1bis
r7	E5	E15	T6;T9;T14	T9:p	E12;E22	r8;r9;r2;r4;r12
r8	E5	E17	T6;T9;T10;T12;T15	T10:m;T9:m;T12:p	E12;E24	r9;r14;r6;r7;r3;r5bis;r1bis;r12;r13;r4;r2;r11;r10
r9	E5	E19	T6;T9;T10;T12;T11;T16	T10:m;T9:m;T11:m;T12:m	E12;E26	r5;r6;r3;r8;r7;r1bis;r11;r13;r10;r14;r2;r5bis;r1;r4;r12
				High		
id	**src**	**dst**	**path**	**points**	**signals**	**conflicts**
r1	E1	E19	T4;T7;T13;T11;T16	T7:p;T11:p	E26;E8	r13;r10;r5;r1bis;r11;r2662;r5bis;r6119;r14
r11	E1	E17	T4;T7;T8;T10;T12;T15	T7:m;T12:p;T8:m;T10:p	E24;E8	r1bis;r6117;r3;r10;r14;r6;r13;r2462;r2662;r5bis;r6119;r5;r1
r1bis	E1	E19	T4;T7;T8;T10;T12;T11;T16	T7:m;T12:m;T8:m;T10:p;T11:m	E26;E8	r10;r14;r6117;r5;r3;r6119;r13;r5bis;r2462;r2662;r6;r11;r1
r3	E24	E10	T15;T12;T10;T8;T5	T12:p;T8:p;T10:p	E17;E3	r2462;r6117;r14;r10;r2662;r13;r6;r6119;r5bis;r11;r1bis
r14	E24	E8	T15;T12;T10;T8;T7;T4	T7:m;T12:p;T8:m;T10:p	E1;E17	r10;r2462;r2662;r6117;r5bis;r6119;r6;r13;r5;r3;r1bis;r11;r1
r2462	E24	E62	T15;T12;T10;T62	T12:p;T10:m	E17;E61	r6117;r5bis;r10;r6;r6119;r2662;r13;r3;r14;r1bis;r11
r5	E26	E8	T16;T11;T13;T7;T4	T7:p;T11:p	E1;E19	r6119;r13;r5bis;r10;r2662;r1bis;r1;r14;r11
r13	E26	E10	T16;T11;T12;T10;T8;T5	T12:m;T8:p;T10:p;T11:m	E19;E3	r2662;r6119;r6;r6117;r5bis;r10;r1;r5;r3;r11;r1bis;r2462;r14
r5bis	E26	E8	T16;T11;T12;T10;T8;T7;T4	T7:m;T12:m;T8:m;T10:p;T11:m	E1;E19	r6;r10;r6117;r2662;r6119;r5;r2462;r14;r13;r1bis;r3;r11;r1
r2662	E26	E62	T16;T11;T12;T10;T62	T12:m;T10:m;T11:m	E19;E61	r6;r10;r6119;r6117;r13;r14;r3;r1bis;r2462;r11;r5bis;r1;r5
r6	E3	E17	T5;T8;T10;T12;T15	T12:p;T8:p;T10:p	E10;E24	r6119;r10;r6117;r2662;r5bis;r13;r2462;r3;r11;r14;r1bis
r10	E3	E19	T5;T8;T10;T12;T11;T16	T12:m;T8:p;T10:p;T11:m	E10;E26	r6117;r6119;r14;r1bis;r6;r2462;r2662;r3;r11;r5bis;r1;r13;r5
r6117	E61	E17	T62;T10;T12;T15	T12:p;T10:m	E24;E62	r6119;r3;r11;r2462;r10;r14;r13;r1bis;r6;r5bis;r2662
r6119	E61	E19	T62;T10;T12;T11;T16	T12:m;T10:m;T11:m	E26;E62	r5;r6;r13;r2662;r6117;r14;r10;r2462;r1bis;r3;r11;r5bis;r1
				Low		
id	**src**	**dst**	**path**	**points**	**signals**	**conflicts**
r2	E22	E12	T14;T9;T6	T9:p	E15;E5	r7;r551;r5212
r7	E5	E15	T6;T9;T14	T9:p	E12;E22	r551;r5212;r2
r551	E5	E51	T6;T9;T51	T9:m	E12;E52	r5212;r7;r2
r5212	E52	E12	T51;T9;T6	T9:m	E5;E51	r7;r551;r2

Abstraction of Routes. We will now describe the relationship between the routes in the interlocking table for the ThreeTracksStation Network and the routes in the interlocking tables for the High and Low sub-networks. Figure 3 shows the routes of ThreeTracksStation: actually, for readability, it shows only *up* routes, that is, routes that have as source and destination signals in the up

direction. These routes are shown in different colour and dotting to distinguish those that are fully contained in one of the two sub-networks, and hence are maintained substantially unchanged in either High (e.g. route r1) or in Low (e.g. route r2), and those that go through the cut, that need to be substituted (*abstracted*) by (often fewer) routes both in High and Low: Figure 4 shows for example how route r551 in Low abstracts both r8 and r9 in ThreeTracksStation).

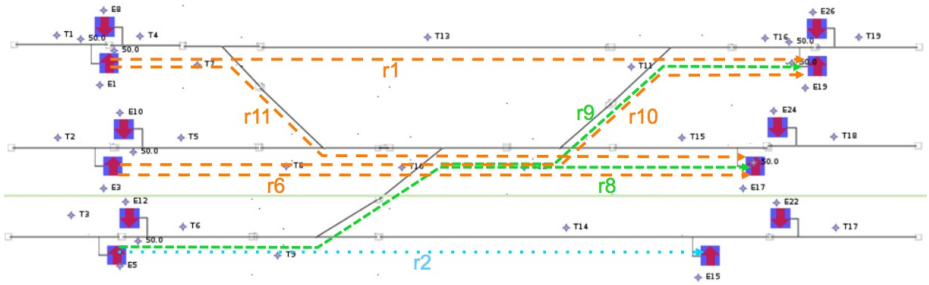

Fig. 3: Up routes of the ThreeTracksStation Network (alternative route r1bis is not shown).

In general, the set $Routes(N)$ of routes of a network N (in our case ThreeTracksStation) is partitioned in three disjoint sets: RH, RL, Th, which are respectively fully contained in Low, in High and passing through the cut. The cut defines two abstraction functions $\gamma_H : Th \to RH'$ and $\gamma_L : Th \to RL'$ that produce the sets of abstract routes for the Low and High networks respectively. The abstraction functions are total and surjective. The routes of the sub-networks

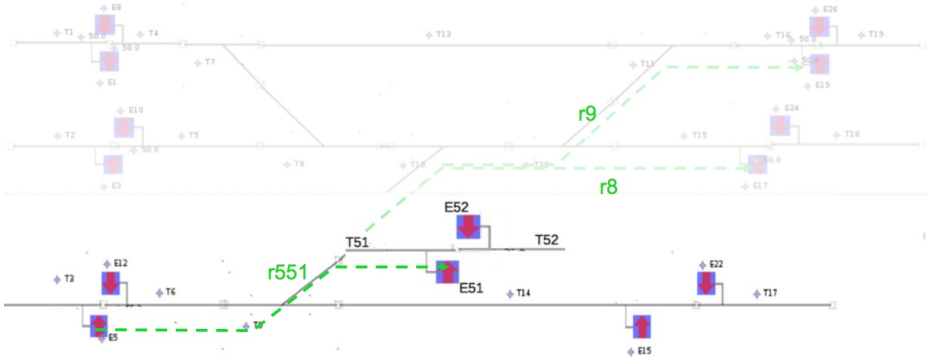

Fig. 4: Abstraction of Routes

are given by (see Fig. 5):
$Routes(Low) = RL \cup RL'$ and $Routes(High) = RH \cup RH'$.

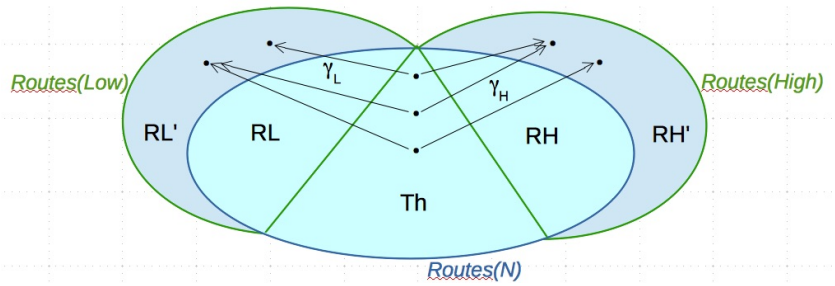

Fig. 5: Relation between the routes of the network and the sub-networks.

Decomposition Relations. Fig. 6 illustrates the decomposition of networks

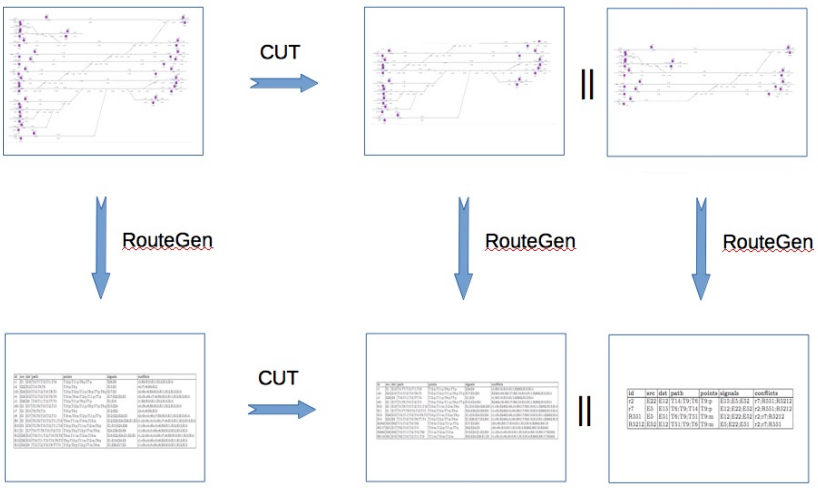

Fig. 6: Decomposition of networks and interlocking tables. The RouteGen arrows represent the interlocking table generator.

and tables. Following the upper "CUT" arrow (representing the horizontal cut operation), the network is decomposed into two sub-networks for which interlocking tables are then generated. These tables provide a decomposition (following the lower "CUT" arrow) of the interlocking table that one would generate from the full network when not using the compositional approach.

The following rules define the relationship between the two sub-networks and the full network, as well the relationship between their associated interlocking tables. Rule 1 defines how the sub-networks are created, rules 2-3 and 4-5 define the route abstractions into sets RL' and RH', and rules 6-10 define the path, required point and signal settings, and route conflicts of the routes in the decomposed interlocking tables, in terms of the corresponding data in the full interlocking table. The rules are instantiated for the ThreeTracksStation example. The different sets of elements have as a suffix the name of the sub-network to which they belong: no suffix means the set belongs to the full network.

1. $Linears_{Low} \cup Linears_{High} = Linears, Linears_{Low} \cap Linears_{High} = \emptyset$
   $Points_{Low} \cup Points_{High} = Points, Points_{Low} \cap Points_{High} = \emptyset$
   $Signals_{Low} \cup Signals_{High} = Signals, Signals_{Low} \cap Signals_{High} = \emptyset$
   The sub-networks Low and High are actually built, respectively, over the sets of elements:
   $(Linears_{Low} \cup \{T51, T52\}, Points_{Low}, Signals_{Low} \cup \{E51, E52\})$
   $(Linears_{High} \cup \{T61, T62\}, Points_{High}, Signals_{High} \cup \{E61, E62\})$
2. all the *up routes* in the ThreeTracksStation starting from $s \in Signals_{Low}$ that incur in the cut (that is, in the path have the pair T9,T10, in our case) are abstracted in the Low sub-network by a route going from s to the stub signal E51;
3. all the *down routes* in the ThreeTracksStation arriving to $s \in Signals_{Low}$ that incur in the cut (that is, in the path have the pair T10,T9) are abstracted in the Low sub-network by a route going from the stub signal E52 to s;
4. all the *down routes* in the ThreeTracksStation starting from $s \in Signals_{High}$ that incur in the cut (that is, in the path have the pair T10,T9) are abstracted in the High sub-network by a route going from s to the stub signal E62;
5. all the *up routes* in the ThreeTracksStation arriving to $s \in Signals_{High}$ that incur in the cut (that is, in the path have the pair T9,T10) are abstracted in the High sub-network by a route going from the stub signal E61 to s;
6. the *path* of the abstract routes contains only the elements to (from) the cut, plus the added stub elements as destination (source); the path of the other routes is unchanged.
7. the *points* of the abstract routes include only the points on the path to (from) the cut; the points of the other routes are unchanged.
8. each abstract route in the Low (High) sub-network abstracting a route r in the full network, keeps as its *signals* that signal of r that was placed in Low (High), while the signal placed on the opposite side of the cut in High (Low) is replaced by the stub signal contrary to the direction of the route. The signals of the other routes are unchanged, except the cases where a signal is placed on the opposite side of the cut, in which case it is replaced by the stub signal in opposite direction of the route.
9. the *conflicts* of an abstract route $r \in RL'$ (RH') in the Low (High) sub-network are given by:
   (a) all the maintained routes of Low (High) that are in conflict in ThreeTracksStation with any of the routes abstracted by r;

(b) all the abstract routes of Low (High) that abstract routes in ThreeTracksStation in conflict with any of the routes abstracted by r;

10. the *conflicts* of a route $r \in RL$ (RH) in the Low (High) sub-network preserved in the cut are given by:
    (a) all the maintained routes of Low (High) that are in conflict in ThreeTracksStation with r;
    (b) all the abstract routes of Low (High) that abstract routes in ThreeTracksStation that are in conflict with r;

These rules can be easily generalised to the "horizontal" cut of any network, including the special case of a network where the cut is on up routes that go from the high part to the low part.

4.3 Safety Verification

In the third step, after having generated the interlocking tables, the safety verification is performed for the sub-networks and their associated interlocking tables, using the RobustRailS verification tools.[8]

Table 2 shows metrics, for the separate verification of the Low, the High and the ThreeTracksStation networks, respectively. Furthermore, the row "Low + High" shows metrics for the combined compositional verification: the execution time is the sum of the execution times of Low and High, and the memory usage is the maximum of the memory usages of Low and High. Time is measured in seconds and memory in MB. As expected, the results show that the compositional approach is advantageous both in terms of the time and memory usage.

Table 2: Verification metrics for the ThreeTracksStation case study.

	Linears	Points	Signals	Routes	$log_{10}(\|S\|)$	Time	Memory
Low	6	1	6	4	30.50	3.37	93.8
High	11	5	10	12	77.11	187.26	528.1
Low + High						190.63	528.1
ThreeTracksStation	13	6	12	14	91.31	292.68	698.3

4.4 Soundness of the Approach

The soundness of the compositional approach amounts to the following theorem.

Theorem 1. *Given a Network N and its sub-networks H and L obtained by a horizontal cut, if H and L are separately verified to satisfy safety, then N satisfies safety too.*

[8] The verifications were performed on a machine with an Intel(R) Xeon(R) CPU E5-2667 0 @ 2.90GHz, 64GB RAM, CentOS 6.6, Linux 2.6.32-504.8.1.el6.x86_64 kernel.

The above considerations on abstracted routes are the ground on which to base a proof. A formal proof can be made in a similar way to the proof presented in [13]: safety properties are expressed as universal quantifications over the sets of linear/point sections, hence, assuming safety is proved for each section of both sub-networks, we need to prove safety for each section in the original network. This can be done by case analysis for three cases: for sections only involved in routes in RL, sections only involved in routes in RH, and sections involved by through routes in Th. The first two cases immediately follows from the assumption. In the third case, the state of some sections (say, in Low) is actually related to the state of the stub added to the Low sub-network, that abstracts the state of sections of High that belong to routes in Th in the full network. We then reason by contradiction, assuming that the state of some of the latter sections violates a safety property for the full network, while their abstraction in the stub of Low does not violate any property in Low. We show that such sections violate safety for High as well, contradicting the initial hypothesis that safety is proved for the High network.

5 A more Complex Example

The next example (called Fismn) is inspired by the layout of the main station of Florence in Italy, and actually refers to a portion roughly of the size of a quarter of the entire station. The layout shown in Fig.7 has been recovered from Google Maps, so there is no actual relation between this layout and any implemented real interlocking system. Nevertheless, this layout realistically represents a feature that can be found in many large stations, that is a route that traverses all the other routes for connecting the lower incoming track to the upper exiting track. In the real Florence station, the next portion of the layout at the right of this includes the reverse traversing route.

For the Fismn network, we applied the compositional verification approach. In Fig.7 the dotted green horizontal line represents the operated cut, between the T30 and T31 points. Note that this cut lies on the low-to-high traversing path. Also in this case the compositional verification is advantageous for the time and memory requirements (see Table 3). Notice that the memory consumption of the Fismn network is actually close to the memory limits (64 GB) so it can be predicted that a bit larger model cannot be treated unless the compositional approach is adopted.

Table 3: Verification metrics for the Fismn case study.

	Linears	Points	Signals	Routes	$log_{10}(\|S\|)$	Time	Memory
Low	28	13	26	56	243.04	12895.35	12176.6
High	25	10	24	66	239.07	8052.92	9517.9
Low + High						20948.27	12176.6
Fismn	49	23	46	124	472.93	51770.64	42483.7

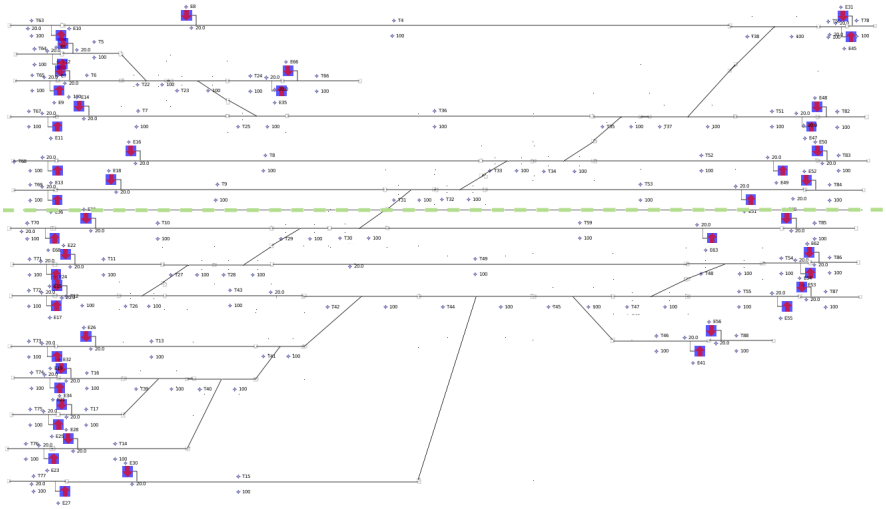

Fig. 7: The Fismn example layout.

6 Conclusions

We have presented a compositional approach to the problem of model checking interlocking systems of large railway stations. The approach builds up over previous work, by proposing a more general way of decomposing a station layout, that has successfully been applied to a large portion of a real world station. The approach achieves significant improvements in verification time and memory usage, taking into account that the aim of the proposed decomposition is to be able to keep time and memory resources needed for the verification within feasible limits. In fact the idea is to apply multiple cuts, in order to chop a large network in tractable chunks, each to be verified separately. This is the main direction of future work, which will need generalising the decomposition process and automating it by means of a tool supporting the network cutting process. In parallel to this effort to provide an automatic verification method for large networks, some other investigation is still needed, e.g. investigating how counterexamples of a safety verification of the sub-networks carry over to counter examples of the full network, or extending the approach to deal with peculiar interlocking safety functions, such as flank protection, or overlap, that have not yet been considered for the horizontal cut.

Acknowledgement

The authors would like to express their gratitude to Jan Peleska and Linh Hong Vu with whom Anne Haxthausen developed the RobustRailS verification method and tools used in the presented work.

References

1. Andrea Bonacchi, Alessandro Fantechi, Stefano Bacherini, and Matteo Tempestini. Validation process for railway interlocking systems. *Sci. Comput. Program.*, 128:2–21, 2016.
2. CENELEC European Committee for Electrotechnical Standardization. *EN 50128:2011 – Railway applications – Communications, signalling and processing systems – Software for railway control and protection systems*. 2011.
3. Alessio Ferrari, Gianluca Magnani, Daniele Grasso, and Alessandro Fantechi. Model Checking Interlocking Control Tables. In Eckehard Schnieder and Géza Tarnai, editors, *FORMS/FORMAT 2010 – Formal Methods for Automation and Safety in Railway and Automotive Systems*, pages 107–115. Springer, 2010.
4. Helle Hvid Hansen, Jeroen Ketema, Bas Luttik, Mohammad Reza Mousavi, Jaco van de Pol, and Osmar Marchi dos Santos. Automated Verification of Executable UML Models. In Bernhard K. Aichernig, Frank S. de Boer, and Marcello M. Bonsangue, editors, *Formal Methods for Components and Objects*, volume 6957 of *Lecture Notes in Computer Science*, pages 225–250. Springer, 2010.
5. Anne E. Haxthausen, Marie Le Bliguet, and Andreas A. Kjær. Modelling and Verification of Relay Interlocking Systems. In Christine Choppy and Oleg Sokolsky, editors, *Foundations of Computer Software, Future Trends and Techniques for Development*, volume 6028 of *Lecture Notes in Computer Science*, pages 141–153. Springer, 2010.
6. Anne E. Haxthausen, Jan Peleska, and Sebastian Kinder. A Formal Approach for the Construction and Verification of Railway Control Systems. *Formal Aspects of Computing*, 23(2):191–219, 2011.
7. Anne E. Haxthausen, Jan Peleska, and Ralf Pinger. Applied Bounded Model Checking for Interlocking System Designs. In Steve Counsell and Manuel Núñez, editors, *Software Engineering and Formal Methods - SEFM 2013 Collocated Workshops. Revised Selected Papers*, volume 8368 of *Lecture Notes in Computer Science*, pages 205–220. Springer, 2014.
8. Philip James, Faron Möller, Hoang Nga Nguyen, Markus Roggenbach, Steve Schneider, and Helen Treharne. Decomposing scheme plans to manage verification complexity. In *FORMS/FORMAT 2014 - 10th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems*, pages 210–220. Institute for Traffic Safety and Automation Engineering, Technische Universität Braunschweig, 2014.
9. Phillip James, Andy Lawrence, Faron Moller, Markus Roggenbach, Monika Seisenberger, Anton Setzer, Karim Kanso, and Simon Chadwick. Verification of solid state interlocking programs. In Steve Counsell and Manuel Núñez, editors, *Software Engineering and Formal Methods - SEFM 2013 Collocated Workshops. Revised Selected Papers*, volume 8368 of *Lecture Notes in Computer Science*, pages 253–268. Springer, 2014.
10. Phillip James, Faron Moller, Hoang Nga Nguyen, Markus Roggenbach, Steve Schneider, and Helen Treharne. Techniques for modelling and verifying railway interlockings. *International Journal on Software Tools for Technology Transfer*, 16(6):685–711, 2014.
11. Christophe Limbrée, Quentin Pecheur, Charles Moller, and Stefano Tonetta. Verification of railway interlocking - compositional approach with OCRA. In Thierry Lecomte, Ralf Pinger, and Alexander Romanovsky, editors, *Reliability, Safety and Security of Railway Systems - RSSR 2016*, volume 9707 of *Lecture Notes in Computer Science*. Springer, 2016.

12. Hugo D. Macedo, Anne E. Haxthausen, and Alessandro Fantechi. Compositional verification of multi-station interlocking systems. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation*, volume 9953 of *Lecture Notes in Computer Science*. Springer, 2016.
13. Hugo Daniel Macedo, Alessandro Fantechi, and Anne E. Haxthausen. Compositional model checking of interlocking systems for lines with multiple stations. In Clark Barrett, Misty Davies, and Temesghen Kahsai, editors, *NASA Formal Methods: 9th International Symposium, NFM 2017, Proceedings*, pages 146–162. Springer International Publishing, 2017.
14. Jan Peleska. Industrial-Strength Model-Based Testing - State of the Art and Current Challenges. In Alexander K. Petrenko and Holger Schlingloff, editors, 8th Workshop on Model-Based Testing, Rome, Italy, volume 111 of *Electronic Proceedings in Theoretical Computer Science*, pages 3–28. Open Publishing Association, 2013.
15. Gregor Theeg, Sergeĭ Valentinovich Vlasenko, and Enrico Anders. *Railway Signalling & Interlocking: International Compendium*. Eurailpress, 2009.
16. Verified Systems International GmbH. *RT-Tester Model-Based Test Case and Test Data Generator - RTT-MBT - User Manual*, 2013. Available on request from `http://www.verified.de`.
17. Linh H. Vu, Anne E. Haxthausen, and Jan Peleska. A Domain-Specific Language for Railway Interlocking Systems. In Eckehard Schnieder and Géza Tarnai, editors, *FORMS/FORMAT 2014 - 10th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems*, pages 200–209. Institute for Traffic Safety and Automation Engineering, Technische Universität Braunschweig, 2014.
18. Linh H. Vu, Anne E. Haxthausen, and Jan Peleska. Formal modeling and verification of interlocking systems featuring sequential release. In Cyrille Artho and Peter Csaba Ölveczky, editors, *Formal Techniques for Safety-Critical Systems*, volume 476 of *Communications in Computer and Information Science*, pages 223–238. Springer International Publishing, 2015.
19. Linh Hong Vu. *Formal Development and Verification of Railway Control Systems - In the context of ERTMS/ETCS Level 2*. PhD thesis, Technical University of Denmark, DTU Compute, 2015.
20. Linh Hong Vu, Anne E. Haxthausen, and Jan Peleska. Formal modelling and verification of interlocking systems featuring sequential release. *Science of Computer Programming*, 133, Part 2:91 – 115, 2017. http://dx.doi.org/10.1016/j.scico.2016.05.010.
21. K. Winter. Symbolic Model Checking for Interlocking Systems. In Francesco Flammini, editor, *Railway safety, reliability, and security: technologies and systems engineering*. IGI Global, 2012.
22. Kirsten Winter. Optimising Ordering Strategies for Symbolic Model Checking of Railway Interlockings. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies*, volume 7610 of *Lecture Notes in Computer Science*, pages 246–260. Springer, 2012.